Windows Forensics Cheat Sheet (Windows 7/10/11)

Nikolaos Kranidiotis • osec.gr

Quick Triage

- `query user` – Show who is currently logged on (local console or RDP sessions). Identify any unexpected active user sessions (could be an intruder).
- `wevtutil qe Security /c:20 /rd:true /f:text` – Quickly display the last 20 Security events (reverse chronological) on-host for rapid triage.
- `net user` & `net localgroup administrators` – List all local accounts and admin group members. Look for unauthorized or recently created accounts (especially in Administrators group).
- `tasklist /v` – List running processes with verbose info. Look for odd process names, very high PIDs (new processes), or ones running

from unusual paths (e.g. user Temp or AppData).

- `netstat -ano` – Show all open ports and network connections with associated PIDs. Investigate any unknown external IP connections or strange listening ports (possible backdoors).
- Quick log scan: Use Event Viewer or `wevtutil` to review recent Security log entries. Look for multiple failed logons (ID 4625) followed by a success (4624), new account creation (4720), or privilege escalation (4672) around the incident. Also check for Event 1102 (audit log cleared – often attacker activity).

Suspicious Artifacts Locations

- **Hidden AppData & ProgramData:** Inspect `C:\Users\<User>\AppData\` (Local/Roaming) and `C:\ProgramData` for odd executables or scripts. These directories are hidden by default and frequently abused by malware.
- **Temp directories:** Check `C:\Windows\Temp` and user temp (e.g. `C:\Users\<User>\AppData\Local\Temp`) for malicious files or archives. Attackers often drop payloads or stage data in temp folders (large *.zip or *.rar files here are suspicious).
- **Recycle Bin:** Look in `C:\$Recycle.Bin\` (per user SID) for unexpected files. Malware can hide in the Recycle Bin and even run from it, so recover and examine "deleted" items.
- **System folders:** Scrutinize `C:\Windows\System32` and `C:\Windows\SysWOW64` for any files that don't belong (especially unsigned EXEs/DLLs with odd names). Attackers may place trojans here to blend in with legitimate system files.
- **User directories:** Review common user folders (Desktop, Documents, Downloads) for rogue programs or scripts disguised among normal files. Sometimes attackers store tools or data dumps in these locations to avoid suspicion.

Registry Hives & Key Artifacts

- **SAM:** `C:\Windows\System32\config\SAM` (HKLM\SAM) – contains local user account data and password hashes. Check for suspicious new user accounts or unexpected password changes. Hashes can be extracted for offline cracking.
- **SYSTEM:** `C:\Windows\System32\config\SYSTEM` (HKLM\SYSTEM) – records system configuration. Key forensic data includes the *ShimCache*

(AppCompatCache) of executed files, USB device history (USBSTOR with first/last connection times), installed services/drivers info, network interfaces, and the computer's timezone and last shutdown time.

- **SOFTWARE:** `C:\Windows\System32\config\SOFTWARE` (HKLM\SOFTWARE) – lists installed programs and OS settings. Contains autorun entries (Run/RunOnce keys for all users), network profiles, and Scheduled Tasks definitions. Also see `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList` for user SIDs mapped to profile paths.
- **SECURITY:** `C:\Windows\System32\config\SECURITY` (HKLM\SECURITY) – stores local security policy and *LSA Secrets* (cached domain creds, service account passwords). Review audit policy settings here (weak auditing means fewer log events) and consider dumping LSA secrets (e.g., via `mimikatz`) to reveal stored credentials.
- **NTUSER.DAT (per user):** `C:\Users\<User>\NTUSER.DAT` (HKU\ *SID*) – each user's registry hive. Contains user-specific artifacts like *UserAssist* (GUI program execution history, ROT13 encoded), *RecentDocs* and *RunMRU* (recent files and Run dialog commands), user-level *Run* autorun entries, and shellbag data for folders accessed by the user.
- **UsrClass.dat (per user):** `C:\Users\<User>\AppData\Local\Microsoft\Windows\UsrClass.dat` (HKU\ *SID* _Classes) – user's shell-related settings. Contains *ShellBag* info (folders browsed, including on external drives) and *MUICache* (last used GUI app names), which can indicate files/folders a user (or attacker using that account) accessed.
- **Amcache.hve:** `C:\Windows\AppCompat\Programs\Amcache.hve` – records execution metadata for programs (e.g. file path, SHA-1 hash, first/last run time) on Windows 8+ systems. Use Amcache to find evidence of program execution (even of programs that have been deleted or renamed).
- *MUICache precision:* On modern Windows (Vista→11) MUICache resides under `HKCU\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache` (in `UsrClass.dat`). Legacy XP used `HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache` (in `NTUSER.DAT`).
- *ShellBags:* Artifact data is split across `NTUSER.DAT` and `UsrClass.dat` on modern versions; analyze both for a complete folder access history.

Persistence & Autostart

- **Services & Drivers:** List auto-start services (`services.msc` or `Get-Service | ? StartType -eq 'Automatic'`) and look for unfamiliar names or paths. Malicious services often have random names or point to executables in Temp or ProgramData. A new service installation triggers event ID 7045 in the System log. Check `HKLM\SYSTEM\CurrentControlSet\Services` for odd entries (and their `ImagePath`).
- **Scheduled Tasks:** Run `schtasks /query /FO LIST /V` to enumerate scheduled tasks. Investigate tasks with weird names or those

launching from non-standard locations (e.g., a task running an EXE under AppData). Security event IDs **4698** (created), **4699** (deleted), **4700** (enabled), **4701** (disabled), **4702** (updated) record Scheduled Task changes. *Note:* **4697** is "a service was installed.". Task files are stored under `C:\Windows\System32\Tasks\` ; examine any suspicious XML tasks for hidden actions.

- **Registry Run Keys:** Check autoruns in `HKLM\Software\Microsoft\Windows\CurrentVersion\Run` (and `…\RunOnce`) as well as the per-user `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` . Attackers commonly use these to persist malware at startup – look for entries pointing to unknown executables or scripts.
- **Startup Folders:** Inspect the Startup folder for each user (`%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup`) for unauthorized files or shortcuts. Anything placed here will run at user login. Also check the All Users startup at `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp` .
- **WMI Persistence:** Advanced attackers may use WMI event subscriptions to execute payloads. Use PowerShell to query `Get-WmiObject __EventFilter` , __FilterToConsumerBinding, and __CommandLineEventConsumer for odd entries (e.g., scripts or commands that launch on a schedule or event).
- **Autoruns Tool:** Use Sysinternals `Autoruns` to automatically enumerate all auto-start extensibility points (services, tasks, Run keys, startup folder, browser helper objects, etc.). Autoruns highlights unsigned or unknown entries – a quick way to spot persistent malware.

Logs & Event History

- **Security Log:** Review authentication and audit events. Key IDs: 4624 (successful login), 4625 (failed login), 4672 (admin logon with privileges), 4720 (user account created), 4732 (member added to a local group such as Administrators), 4740 (account locked). Multiple 4625 failures followed by a 4624 could indicate brute force or password guessing. Also check 4648 (explicit credential logon) and 4688 (process start events, if Audit Process Tracking is enabled).
- **System Log:** Check for system-level changes and errors. Event ID 7045 ("A service was installed") is a red flag for malware persistence. Look for driver errors or crashes, unexpected reboots, time changes, and any event around the incident time that might indicate a security control failure (e.g., AV service stopped).
- **Application Log:** See if any applications (especially security tools) logged errors or alerts. Windows Defender, for example, logs detections in either Application or its own Operational log. Investigate any crash reports (they might correspond to exploited processes or malware crashing).
- **Other Logs:** If enabled, check "Applications and Services" logs – e.g., PowerShell Operational log (event 4103/4104 for script execution)

to catch base64-encoded PowerShell commands, or Sysmon if present (for detailed process/file/network events). Also check TerminalServices logs for RDP connection events if relevant.

- **Log Integrity:** Be alert to log tampering. Event ID 1102 in Security indicates the audit log was cleared. Also look for Event 104 (Log service shutdown) or suspicious gaps in the logs. If logs abruptly stop or have a time gap during the incident, the attacker may have cleared them. Ensure log settings (retention, size) weren't altered to suppress evidence.

File Carving & Deleted Files

- **Recycle Bin:** Examine `$Recycle.Bin` for each user – files here can reveal what attackers tried to delete. Use forensic tools or manual recovery to restore these files (the original filename and deletion time can often be recovered from the Recycler info).
- **Volume Shadow Copies:** If Shadow Copies exist (`vssadmin list shadows`), mount them to retrieve previous versions of files or registry hives that were deleted or altered by the attacker. Shadow copies can provide historical data that fills gaps (e.g. an attacker deleted a log, but an older copy still has it).
- **Undelete & Carving:** Utilize file carving on a forensic image of the disk:
  - Use tools like `photorec` or `Scalpel` to carve for common file types (documents, images, executables) from unallocated space.
  - Use The Sleuth Kit (`fls` , `icat`) to list and extract deleted NTFS files by MFT record. For example, `fls -r -m C: image.dd` will show deleted files (marked with *).
- **Timeline via $MFT:** Parse the NTFS Master File Table to include deleted file timestamps in your timeline (many tools can do this, or use `log2timeline.py`). This can reveal when files were deleted or created relative to other events.
- **Imaging First:** Always perform carving and undelete on a forensic image or write-protected disk. Use tools like FTK Imager to create an image (E01/RAW) of the drive before heavy analysis, to avoid altering last-access times or other metadata on the original system.

Memory Acquisition & Analysis

- **Memory Dump Tools:** Acquire RAM from the live system using trusted tools. Options include **FTK Imager** (File > Capture Memory),

DumpIt (one-click portable dump tool), or **Magnet RAM Capturer**. A small free tool is Belkasoft's Live RAM Capturer. Save the dump to an external drive if possible (to avoid overwriting data on the suspect system).

- **Pagefile/Hibernation:** If a full memory dump isn't available, collect `C:\pagefile.sys` and `C:\hiberfil.sys` – they contain memory fragments (and sometimes credentials) from the system state. These can be analyzed similarly to RAM dumps for artifacts.
- **Volatility Framework:** Use `Volatility` (or `Volatility3`) to analyze the memory image:
  - `pslist` / `pstree` – list processes and parent-child relationships (find processes that shouldn't be there or odd PPIDs).
  - `dlllist` / `ldrmodules` – list loaded DLLs for each process (spot injected DLLs or missing module entries).
  - `netscan` – list open connections and listening ports from memory (may catch sockets opened by malware that no longer show in live netstat).
  - `cmdline` / `consoles` – retrieve process command-line arguments and any console input/output (could reveal attacker command history).
  - `hivelist` / `dumpregistry` – locate registry hives in memory; allows extracting SAM, SECURITY, etc., directly from the RAM image.
  - `malfind` – scan for hidden/injected code in process memory (identifies suspicious memory segments with RWX permissions).

  Example: `volatility -f memory.bin --profile=Win10x64_19041 pslist` (Volatility 2) or `vol.py -f memory.bin windows.pslist` (Volatility 3) to list processes in the dump.
- **Other Tools:** Try `Rekall` (memory analysis fork of Volatility) or tools like `MemProcFS` which mount a memory image as a virtual file system for easy exploration. These can complement Volatility for specific analysis needs.
- **Memory Analysis Tips:** Look for discrepancies between memory and disk: processes present in memory that have no file on disk (indicative of fileless malware), suspicious DLLs loaded in normal processes, or plaintext secrets. Dump suspicious processes (`volatility procdump`) for further analysis with AV or reverse engineering. Also search the memory image for keywords like "password" or known IoCs – credentials and even malicious URLs often linger in RAM.

Timeline Reconstruction

- **File System Timeline:** Leverage the $MFT timestamps to build a timeline of file events. For example, use `MFTECmd` to parse the Master File Table or `fls` / `mactime` (Sleuth Kit) to create a CSV timeline. Include MACB times for files to correlate creations/modifications with suspect activity.

- **log2timeline/Plaso:** Use `log2timeline.py` (Plaso) on an image to collect a comprehensive timeline from many sources (MFT, event logs, registry hives, browser history, etc.). Then use `psort.py` to filter/sort events. This "super-timeline" can help uncover sequences of attacker actions across the system.
- **Correlate Events:** Manually correlate relevant logs and artifacts around key times. E.g., if malware was executed at 14:30, check Security log around 14:30 for new logon or privilege use, check file timestamps for files created/modified at 14:30, and check Task Scheduler log for any tasks running at that time. Aligning these can recreate an attack narrative.
- **Visualization:** For complex timelines, use tools like Timesketch or ELK to visualize spikes or clusters of activity. Clusters of events in short succession (file mods, new processes, log entries) often indicate malicious activity that can stand out from normal baseline.
- **Timestamps Caveat:** Keep in mind attackers may manipulate timestamps (timestomping) or some file ops preserve timestamps (copying a file retains original timestamps). Always verify timeline findings with multiple artifacts (e.g., Prefetch last run times, log entries) to ensure accuracy.

Network & Exfiltration

- **Active Connections:** Use `netstat -ano` (or Sysinternals TCPView) to identify current network connections and listening ports. Unknown external IPs or ports can indicate C2 or exfiltration channels. Cross-reference PIDs with `tasklist /svc` to see which process/service is responsible.
- **Firewall/Proxy Logs:** Check Windows Firewall logs (if enabled) at `C:\Windows\System32\LogFiles\Firewall\pfirewall.log` for unusual blocked or outbound traffic around the incident. Also, if the host uses a proxy or EDR that logs connections, review those for anomalies (e.g., large uploads, connections to new domains).
- **Data Staging:** Look for signs of data being gathered before exfiltration: large archive files (e.g. .zip, .7z) created in temp or user folders, or tooling like `robocopy` logs (attackers sometimes use it to collect files). The presence of compression tools or usage (like RAR, 7zip) on the system is a clue.
- **Transfer Tools:** Check for usage of built-in networking tools. Examples: `certutil -urlcache -f` (file download), `bitsadmin` (might list jobs used for transfer), `PowerShell Invoke-WebRequest` or `curl.exe` . Also examine any scripts or batch files for commands like `ftp.exe` or `powershell -Enc` (which could be a file download/upload in encoded form).
- **External Destinations:** Identify where data might have gone. Review DNS cache (`ipconfig /displaydns`) for odd domains. If RDP was used, consider that clipboard or drive redirection might have been enabled for file transfer (though not logged by default). Look at cloud

storage usage (OneDrive/Dropbox folders) for any recently added files if those were available to the user.

- **Packet Capture (live):** If you are live on the system and suspect ongoing exfil, you can capture traffic using built-in `pktmon` or `netsh trace start`. However, use with caution on production systems. In many cases, host artifacts and logs will be more readily available than live packet captures on an endpoint.

Malware Hunting & IOCs

- **Filesystem Search:** Search for common attacker traces:
  - Encoded scripts: e.g. find base64 blobs in PowerShell scripts or registry (`-enc` in PowerShell command lines). Many malicious scripts use Base64; decode any suspicious blobs.
  - Known malware keywords: search disk for strings like "mimikatz", "pwdump", "evil" etc., or for file names attackers commonly use (e.g., `logon.ps1`, `rshell.exe`).
  - Unusual file placements: scan for executables in locations they shouldn't be (e.g., .exe in `C:\Windows\` Temp, or a DLL in user Documents folder).
- **Hash & Reputation:** Generate hashes (MD5/SHA256) for unknown binaries and check them against VirusTotal or threat intel databases. Even if not flagged, a hash search might reveal if the file has been seen elsewhere. Be mindful of OPSEC if uploading to VT (it can alert adversaries).
- **Binary Analysis Clues:** Run `strings` on suspicious binaries to look for IP addresses, URLs, or hidden messages. Check the binary's properties: is it digitally signed (and by whom)? What is the compilation timestamp (does it roughly align with the OS or is it very recent)? Packed malware might have telltale strings like "UPX". Use tools like DIE or ExeInfo to detect packers.
- **Rootkit Checks:** If you suspect a kernel/rootkit:
  - Compare different outputs to find hidden processes (e.g., run `tasklist` vs. Volatility's `pslist`). Hidden drivers might appear in registry (`HKLM\SYSTEM\CurrentControlSet\Services`) but not in `driverquery`.
  - Look at autostart locations that load early: e.g., `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\\AppInit_DLLs` or boot execute keys. Attackers may use these for injection.
  - Consider running anti-rootkit tools (GMER, TDSSKiller) on the system as a quick scan, but preferably after imaging, as they can be invasive.

- **Persistence Re-check:** Double back to persistence locations after initial cleanup – some malware installs multiple mechanisms. For example, re-check Scheduled Tasks and Services for any that reappear or were missed (attackers may name them subtly). Also ensure things like WMI subscriptions are checked (they might not be obvious on first pass).
- **User Activity Artifacts:** Investigate what the attacker did on the system:
  - Shortcut files: Check `C:\Users\<User>\Recent\` for LNK files pointing to tools or documents the attacker opened.
  - Jump Lists: Inspect `%APPDATA%\Microsoft\Windows\Recent\AutomaticDestinations\` for jump list files – these can reveal recently accessed files and folders by the attacker's account.
  - Shellbags: Analyze shellbag data (can use RedRipper or Registry Explorer) to see folders accessed (e.g., if the attacker browsed to a sensitive directory, a shellbag entry remains).
  - PowerShell History: Check `%UserProfile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt` for any attacker commands (unless they cleared it).
- **Indicators in Logs:** Look for patterns in Event Logs that suggest malicious activity: e.g., a surge of failed logons (4625) followed by a new login (4624) from the same source, or a Service Control event (7040/7045) where a service startup type was changed or new service installed. Check Windows Defender logs (Event ID 1116 in Windows Defender log for malware detection) and any Application Error events around the time of compromise. Use log queries or SIEM if available to quickly flag these events.
- **YARA & Threat Hunting:** Employ YARA rules to scan memory dumps or suspicious files for known malware families or strings. You can also use Sigma rules on event logs for known attacker behaviors. Tools like `Loki` or `Thor` (by Nextron) can automate IOC scanning on an endpoint with a set of common YARA/Sigma rules.

Credential & Sensitive Data

- **SAM Hashes:** The SAM registry hive contains NTLM password hashes for local accounts. Attackers dumping SAM (via `reg save` or Volume Shadow Copy) can crack these offline. Ensure no new local accounts were created (Security event 4720) and no existing accounts were added to Administrators (4728).
- **Domain Credentials:** If the host is domain-joined, check for cached domain logon hashes (in SECURITY hive's NL$_* entries) and Kerberos tickets. Attackers with admin rights often dump these via `mimikatz` (e.g., `sekurlsa::logonpasswords`) to get cleartext passwords and Kerberos TGTs. The SECURITY hive's LSA Secrets also store things like DefaultDomainName and cached creds.

- **LSASS Process Memory:** The LSASS process holds user credentials in memory. If Credential Guard is not enabled, LSASS may contain plaintext passwords or Kerberos tickets. Check if a dump of LSASS was taken (Event 4661 with LSASS handle, or Sysmon detecting a process access to LSASS). If you have a memory dump, run mimikatz or similar on it to extract any credentials.
- **Windows Credential Manager:** List saved credentials with `cmdkey /list`. Attackers may extract saved RDP/SMB passwords stored here. DPAPI protects them, but if the user's profile is compromised, DPAPI master keys can be used to decrypt these secrets.
- **Browser & Application Passwords:** Check for stored web credentials: e.g., Chrome/Edge store passwords in `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data` (SQLite DB), Firefox in `logins.json` (with `key4.db`). Attackers often run tools (like Nirsoft's utilities) to steal these if available. Also consider email client configs (Outlook PSTs won't store passwords in plaintext, but Thunderbird, etc. might have them saved) and VPN client configs.
- **API/Cloud Keys:** Search for cloud service credentials in config files: e.g., AWS CLI credentials in `%USERPROFILE%\.aws\credentials`, Azure CLI tokens, GCP service account keys, or config files for CI/CD tools. Such keys could allow attackers to pivot to cloud resources – treat them as compromised if found on an owned host.
- **Confidential Documents:** Identify sensitive documents (financial data, password lists, etc.) that the attacker may have accessed or exfiltrated. Use file access timestamps (and shellbags/jump lists) to see if those files were opened. If so, assume their contents are known to the attacker and respond accordingly (e.g., password rotations, notifying affected parties).
- **Encryption Keys:** If the system used encryption (BitLocker, PGP, etc.), determine if keys were accessible. BitLocker keys might be in AD or in the user's Microsoft account if linked; PGP keyrings might be in Documents. An attacker with admin access could extract private keys from certificate stores or key files, so inventory any found and assume they're compromised.

Hidden & Temp Files

- **Hidden Files & Folders:** Attackers may set files/folders with Hidden and System attributes to hide them from normal view. Enable "show hidden files" or use `dir /a` to reveal them. Look for strange folder names (Like "." or using Unicode characters). According to MITRE, adversaries use hidden files to evade detection, so ensure none of those exist in key directories (Windows, ProgramData, etc.).
- **Alternate Data Streams (ADS):** NTFS ADS can hide malicious data behind legitimate files (e.g., `legit.docx:secret.exe`). Use `dir /R` or Sysinternals `streams.exe` to find ADS. If suspicious ADS are found, copy them out (e.g., `more < file.txt:malware.exe > malware.exe`) and analyze. Also check for execution via `wmic process call create "file.txt:malware.exe"` in logs or ShimCache.
- **Cover-Up Tools:** Check if anti-forensic tools were used. For instance, did the attacker run a cleaner like *CCleaner* or *SDelete*?

Prefetch files (in `C:\Windows\Prefetch\*.pf`) for CCleaner, SDelete, etc., will show last run times. Their usage suggests log or file wiping. Also look at Event log 1102 (Security log cleared) as mentioned, and any clearing of application logs.

- **Residual Dumps:** Look for crash dumps or backup files. Windows Error Reporting (WER) dumps in `C:\ProgramData\Microsoft\Windows\WER\` might contain memory of crashed processes (maybe the malware). Also, if the system BSOD'd (check System log), a `MEMORY.DMP` could exist. These dumps can sometimes be analyzed to extract malware or credentials post-incident.
- **Fileless Malware Signs:** Fileless malware lives in RAM, but may leave indirect clues. For example, a running process with no corresponding file on disk (or one that appears and disappears) is suspect. If a process image path in tools like Process Explorer shows as "`(deleted)`", that's a red flag (on Linux this is common, on Windows it might show as missing file). Memory analysis is essential to fully discover fileless malware. Also check the registry for unusual persistence that would reload a fileless payload (like a WMI consumer or Run key with a PowerShell one-liner loading from memory).